Particle Swarm Optimization para Problemas de Secuenciamiento: Total Weighted Tardiness

Leticia Cagnina, Susana Esquivel, Raúl Gallard

Laboratorio de Investigación y Desarrollo en Inteligencia Computacional (LIDIC)[1].
Universidad Nacional de San Luis. Argentina
{lcagnina, esquivel}@unsl.edu.ar

Resumen

Particle Swarm Optimization (PSO) ha sido utilizado exitosamente en diferentes áreas como optimización mutidimensional y multiobjetivo, y entrenamiento de redes neuronales entre otras, pero existen actualmente pocos reportes relacionados a problemas de secuenciamiento. En este trabajo presentamos un PSO modificado (MPSO) que incorpora un método diferente de representación de las partículas y un operador de mutación dinámica similar al utilizado en los algoritmos evolutivos. Este estudio preliminar muestra la performance del algoritmo cuando éste es aplicado a un conjunto de instancias correspondientes al problema de Total Weighted Tardiness, en ambientes de scheduling de máquina única. Este primer estudio muestra a MPSO como un posible método alternativo para resolver problemas de secuenciamiento.

1. Introducción

El algoritmo PSO fue propuesto por James Kennedy y Russell Eberhart [17] en 1995. PSO es una heurística estocástica basada en poblaciones, donde cada partícula representa una solución potencial dentro del espacio de búsqueda y está caracterizada por una posición, una velocidad y un registro de su comportamiento pasado. En cada ciclo de vuelo se evalúa cada partícula para obtener el valor de la función objetivo en su posición corriente. Las partículas "vuelan" a través del espacio de búsqueda influenciadas por 2 factores: a) la mejor posición de la partícula alcanzada hasta el momento, y b) la mejor posición global alcanzada por cualquier partícula de la población.
Diferentes heurísticas [20] como los algoritmos evolutivos y los de colonias de hormigas han sido aplicados exitosamente en la resolución de problemas de *scheduling* [4, 8, 9, 12, 13, 22, 25, 28], aunque no existen trabajos usando el paradigma PSO. Seleccionamos el problema de *scheduling* de *Total Weighted Tardiness* debido a su conocida dificultad en resolverlo.
El objetivo de este trabajo es presentar un algoritmo PSO modificado (denominado MPSO) propuesto para el problema de *scheduling* conocido como *Total Weighted Tardiness* en máquina única, y comparar los resultados preliminares con los valores objetivos obtenidos mediante el uso de heurísticas convencionales y otro enfoque de PSO. El resto del trabajo está organizado de la siguiente manera. La sección 2 describe el problema de *scheduling* tratado. La sección 3, establece los lineamientos del algoritmo que implementa MPSO. El valor de sus parámetros y los resultados obtenidos se muestran en la sección 4. En la sección 5 se presentan las conclusiones preliminares y se bosquejan algunas de las líneas de investigación presente y futuras.

[1] El LIDIC es financiado por la Universidad Nacional de San Luis y la ANPCYT (Agencia Nacional para la Promoción de la Ciencia y la Tecnología)

2. El Problema de Scheduling de Total Weighted Tardiness

En el problema de *scheduling* de *Total Weighted Tardiness* para máquina única [24], *n* jobs se procesan sin interrupción en una única máquina. Ésta sólo puede procesar un único job a la vez. Todo job *j* (*j* = 1, ...,*n*) está disponible para su procesamiento en tiempo 0, requiere un tiempo de procesamiento p_j, tiene un peso positivo w_j, y un tiempo de entrega d_j en el cual, idealmente, el job debería finalizar. Para un orden de procesamiento de jobs establecido, el tiempo de completitud C_j y el *tardiness* $T_j = \max(C_j - d_j, 0)$ del job *j* puede calcularse fácilmente. El problema radica en encontrar el orden de procesamiento de los jobs con el mínimo *total weighted tardiness*

$$\sum_{j=1}^{n} w_j T_j$$

Aunque su formulación es simple, este modelo se traduce en un problema de optimización que es NP-Hard [24].

3. Particle Swarm Optimization para el Problema de Scheduling de Máquina Única

PSO ha sido utilizado exitosamente en diferentes áreas (optimización multidimensional [2, 11, 18, 19, 26, 30], optimización multiobjetivo [7, 15, 23] y entrenamiento de redes neuronales [10, 29] entre otras) aunque existen muy pocos trabajos sobre problemas de secuenciamiento [6, 16].
En esta heurística cada individuo o *partícula* representa una solución potencial dentro del espacio de búsqueda. Cada partícula está caracterizada por una posición, una velocidad que provoca el cambio de dirección de la partícula dentro del espacio de búsqueda, y un registro de su comportamiento pasado. En cada iteración del algoritmo o *ciclo de vuelo*, la función objetivo es evaluada por cada partícula con respecto a su posición corriente, indicando la aptitud o la calidad de la misma. También, en cada ciclo se actualizan las partículas considerando dos valores: *pbest*, el mejor valor alcanzado por la partícula hasta ese momento, y *gbest* el mejor valor encontrado por toda la población. Esta actualización se produce incrementando la velocidad a la posición actual de la partícula:

$$pcurrent_{i,j}^{t+1} = pcurrent_{i,j}^{t} + v_{i,j}^{t+1}$$

siendo $pcurrent^{t}_{i,j}$ la posición *i* de la partícula *j* en el instante de tiempo *t*, y $v^{t+1}_{i,j}$ la velocidad de la misma partícula en el instante posterior, calculada siguiendo la siguiente fórmula:

$$v_{i,j}^{t+1} = wv_{i,j}^{t} + c_1 \text{rand}()(pbest_{i,j}^{t} - pcurrent_{i,j}^{t}) + c_2 \text{rand}()(gbest_{j}^{t} - pcurrent_{i,j}^{t})$$

donde *w* es el peso de inercia (*inertia weight*) [5] cuyo objetivo es controlar el grado de velocidad previa que conservará la partícula. Los coeficientes c_1 y c_2 representan los factores de aprendizaje personal y social respectivamente, y *rand()* es un número aleatorio dentro del rango [0,1].
Nuestro algoritmo MPSO intenta mejorar la performance de HPSO [31], el cual fue propuesto recientemente. HPSO además de utilizar el operador de mutación dinámica, incorpora una representación de *random key* (propuesta por Bean [5]) para la partícula.
En MPSO, las partículas son representadas con permutaciones de jobs válidas, las cuales son evaluadas para determinar su valor objetivo. En MPSO la fórmula de actualización de la velocidad se mantiene igual que en el PSO tradicional anteriormente descripto, aunque sólo se limita a valores positivos. El proceso de actualización de cada partícula en MPSO se realiza de la siguiente manera [16]:

1) Construir el “vector normalizado de velocidades” en el rango [0,1]: dividir cada posición del vector de velocidades por el rango máximo de la partícula. Cada una de estas posiciones representa la *probabilidad* de que esa posición de la partícula sea actualizada.
2) Calcular un valor aleatorio que será usado para determinar, comparándolo con cada valor del vector normalizado, si cada posición de la partícula debe ser actualizada.

Si el valor del vector normalizado es mayor que el número aleatorio entonces se actualiza esa posición, *swap* de la partícula:

3) Obtener el valor que contiene *pbest* en esa posición.
4) Buscar ese valor en la partícula.
5) *Swap* de posiciones de la partícula: Intercambiar la posición donde se encontró ese valor, y la posición en donde se determinó que se requería actualización.

Adoptamos un operador de mutación dinámica para mantener la diversidad de la población. Este operador se aplica para cambiar la partícula con una probabilidad p_m, la cual depende del número total de ciclos de vuelo y el ciclo corriente:

$$pm = max\_pm - \frac{max\_pm - min\_pm}{max\_cycle} current\_cycles$$

De esta forma la mutación se aplicará más frecuentemente al comienzo del proceso de búsqueda y decaerá a medida que el número de ciclos aumenta. La partícula sólo se modifica si el valor objetivo de la partícula mutada es mejor que el valor anterior.

4. Resultados Preliminares

MPSO fue testeado con diez instancias de problemas de 40 jobs, extraídos de la OR-library [21], para el problema de *Weighted Tardiness* para máquina única. Se realizaron 30 corridas para cada instancia. El número máximo de ciclos (*max_cycle*) fue fijado en 10.000. El tamaño de la población fue elegido igual al número de jobs del problema, como sugirió Clerc [6], es decir, 40 partículas. La probabilidad dinámica varía desde 0.1 (*min_pm*) a 0.4 (*max_pm*). Las velocidades máxima y mínima se prefijaron en 1.0 y 0.0 respectivamente, el peso de inercia *w* fue de 1.3 con factores de aprendizaje c_1 y c_2 ambos fijados en 0.3. Estos valores iniciales de parámetros fueron determinados como los mejores valores después de numerosas pruebas. Con ellos se obtuvieron resultados preliminares, conformando así la fase inicial de un estudio en profundidad de la aplicabilidad de PSO a problemas de *scheduling*.
Estos resultados fueron contrastados con cuatro heurísticas convencionales diseñadas específicamente para el problema de *weighted tardiness* (WSPT: *Weighted Shortest Processing Time*, EDD: *Earliest Due Date*, R&M: *Rachamadagu and Morton*, and WH: *Weighted Hodgson*). También se incluyen los resultados obtenidos con HPSO. La Tabla 1 ilustra los resultados, donde la primera columna especifica la instancia considerada, la segunda el *upper-bound* (mejor valor conocido) provisto por la OR-library y las restantes columnas los valores objetivos obtenidos con las correspondientes heurísticas. Los valores en negrita reflejan los mejores de las 6 heurísticas.

Inst	UPP. BND	WSPT	EDD	R&M	WH	HPSO	MPSO
1	913	3070	1590	1150	2250	913	**913**
11	17465	21100	67800	18300	18600	17465	**17465**
21	77774	78300	161000	78000	149000	77785	**77776**
31	6575	20300	19700	6680	10800	6575	**6575**
46	64451	65100	125000	64500	157000	64457	**64453**
56	2099	15600	9340	4340	2890	2099	**2099**
71	90486	93600	145000	90700	183000	90616	**90491**
91	47683	69700	129000	49100	89400	47870	**47797**
101	0	0	0	0	0	0	0
116	46770	64900	101000	48000	123000	**46905**	46991

Tabla 1: Valores objetivos obtenidos con diferentes heurísticas para instancias de 40 jobs.

5. Conclusiones

El algoritmo MPSO fue diseñado para comenzar a estudiar problemas de permutaciones. Para determinar la performance del algoritmo se seleccionó el problema de *scheduling* de *weighted tardiness*, debido a su inherente dificultad. MPSO es un algoritmo PSO modificado con una representación de permutación para las partículas y un operador de mutación dinámica, siendo éstos últimos los únicos cambios realizados al PSO original.

Es importante remarcar que a pesar de su simplicidad algorítmica y de ser un algoritmo de búsqueda ciega, el MPSO es promisorio para encarar problemas de *scheduling* duros.

Actualmente se está trabajando en completar el estudio experimental (para problemas de diferente cantidad de jobs) y en el estudio de MPSO en cuanto a aspectos que lo diferencian de HPSO.

El trabajo futuro apunta a encarar la minimización de la redundancia de los datos como lo propusieron Alba y Chicano [1]. También se estudiará la posibilidad de incluir información adicional al algoritmo (mediante la utilización de buenas semillas: buenas soluciones provistas por buenas heurísticas) para guiar la búsqueda ciega que realiza PSO. La inclusión de vecindarios podría lograr una mejora en la convergencia de este algoritmo(*local model* [18]).

6. Referencias

1. E. Alba, J. F. Chicano, “A Binary Representation for Permutations” Internal report 2004, Universidad de Málaga, Departamento de Lenguajes y Ciencias de la Computación, E.T.S.I Informática, Campus de Teatinos, 29071, Málaga, España.
2. P. J. Angeline, Evolutionary Optimization versus Particle Swarm Optimization: Philosophy and Performance Differences, In V. W. Porto, N.Saravahah, D. Waageh and A. E. Eiben Editors, Evolutionary Programming VII, 7th International Conference EP’98, California, USA, pp. 601-610, Springer Verlag, Lecture Notes in Computer Science N° 1447, March 1998.
3. P. J. Angeline, Using Selection to Improve Particle Swarm Optimization. In International Conference on Evolutionary Computation, Piscataway, NJ., IEEE Service Center, 1998.
4. S. Avci, M. S. Akturk and R. H. Storer, A Problem Space Algorithm for Single Machine Weighted Tardiness Problems, IIE Transactions, Vol. 35, pp. 479-486, 2003.
5. J. C. Bean, Genetic Algorithms and Random Keys for Sequencing and Optimization, ORSA Journal on Computing, Vol. 6, N° 2, Springer 1994, pp. 154-160, 1994.
6. M. Clerc, Discrete Particle Swarm Optimization. Illustred by the Traveling Salesman Problem, 2000. http://www.mauriceclerc.net.
7. C. A. Coello Coello and M. S. Lechuga, MOPSO: A Proposal for Multiobjective Particle Swarm Optimization, In Proceedings of the Congress on Evolutionary Computation, Honolulu, Hawaii, USA, 2002.

8. M. den Besten, T. Stüzle and Marco Dorigo, Ant Colony Optimization for the Total Weighted Tardiness, in Paralell Problem Solving from Nature, PPSN VI International Conference, 1999.
9. M De San Pedro., D Pandolfi., A. Villagra, M. Lasso, G. Vilanova, R. Gallard, "Adding problem-specific knowledge in Evolutionary Algorithms to Solve W-T Scheduling Problems", Proceedings del VIII Congreso Argentino de Ciencias de la Computación, pp343-353, CACIC, Universidad de Buenos Aires 2002.
10. R. C. Eberhart and Y. Shi, Evolving Artificial Neural Networks, In Proceedings of International Conference on Neural Networks and Brain, Beijing, P. R. China, pp. PL5-PL13, 1998.
11. S. C. Esquivel and C. A. Coello Coello, On the Use of Particle Swarm Optimization with Multimodal Functions, Proceedings of The International Congress on Evolutionary Computation, pp 1130-1136, Canberra, Australia, 2003.
12. M. Fedmann and D. Biskup, Single-machine Scheduling for Minimizing Earliness and Tardines Penalties by Meta-heuristic Approaches, Computers and Industrial Engineering, Special Issue: Focused Issue on Applied Meta-heuristics, Vol. 44, Issue 2, pp. 307-323, February 2003.
13. P. F. Franca, A. Mendes and P. Moscato, Memetic Algorithms to Minimize Tardiness on a Single Machine with Sequence-dependent Setup Times, 1999 (http://www.densis.fee.unicamp.br).
14. N. Higashi and I. Hitoshi. , Particle Swarm Optimization with Gaussian Mutation. In Swarm Intelligence Symposium, Indianapolis, Indiana, pp. 72-79, Piscataway, NY., April 2003, IEEE Service Center, April 2003.
15. X. Hu and R. C. Eberhart, Multiobjective Optimization using Dynamic Neighborhood Particle Swarm Optimization, In Swarm Intelligence Symposium, Indianapolis, Indiana, Piscataway, NY., April 2003, IEEE Service Center, April 2003.
16. X. Hu and R. C. Eberhart, Swarm Intelligence for Permutation Optimization: A Case Study of n-Queens Problem, In Swarm Intelligence Symposium, Indianapolis, Indiana, pp. 243-246, Piscataway, NY., April 2003, IEEE Service Center, April 2003.
17. J. Kennedy and R. C. Eberhart, Particle Swarm Optimization, In Proceedings of International Conference on Neural Networks, Perth, Australia, pp. 1942-1948, Piscataway, NY., IEEE Service Center, 1995.
18. J. Kennedy and R. C Eberhart, Swarm Intelligence, Morgan Kaufmann Publishers, California, USA, 2001.
19. J. Kennedy and R. Mendes, Population Structure and Particle Swarm Performance, In Proceedings of International Congress on Evolutionary Computation, Honolulu, Hawaii, Piscataway, NJ., IEEE Service Center, May 2002.
20. T. Morton and D. Pentico, Heuristic Scheduling Systems, Wiley Series in Engineering and Technology Management, John Wiley and Sons, INC, 1993.
21. OR Library, http://mscmga.ms.ic.ac.uk/
22. P. Pandolfi, M. De San Pedro, A. Villagras, G. Vilanova and R. Gallard, Stud Mating Inmigrants in Evolutionary Algorithms to Solve the Earliness-tardiness Scheduling Problems, in Cybernetic and Systems, Taylor and Francis Journal (UK), pp. 391-400, June 2002.
23. K. E. Parsopoulus and M. N. Vrahatis, Particle Swarm Optimization Method in Multiobjective Problems, In Proceedings ACM Symposium on Applied Computing 2002, pp. 603-607, 2002.
24. M. Pinedo, Scheduling: Theory, Algorithms and Systems, First Edition, Prentice Hall, 1995.
25. C. Reeves, A Genetic Algorithm for Flow Shop Sequencing, Computers and Operation Research, Vol. 22, pp. 5-13, 1995.
26. J. Riget and J. S. Vesterstrom, A Diversity-guided Particle Swarm Optimizer – The ARPSO, Technical Report N° 2002-02, Evalife, University of Aarhus, Denmark, 2002.
27. Y. Shi and R. C. Eberhart, A Modified Particle Swarm Optimizer, In International Conference on Evolutionary Computation, Anchorage, AK, Piscataway, NJ., IEEE Service Center, 1998
28. Y. Tsujimura, M. Gen and E. Kubota, Flow Shop Scheduling with Fuzzy Processing Time using Genetic Algorithms, The 11th Fuzzy Systems Symposium, Okinawa, pp. 248-252, 1995.
29. F. van der Bergh and A. P. Engelbrecht, Trainning Product Units Networks using Cooperative Particle Swarm Optimizers, In Proceedings of INNS-IEEE International Join Conference on Neural Networks, Washington DC, USA, 2001.
30. F. van der Bergh, An Analysis of Particle Swarm Optimization, PhD Thesis, University of Pretoria, South Africa, November 2002.
31. L. Cagnina, S. Esquivel y R. Gallard, Particle Swarm Optimization for Sequencing Problems: A Case Study. 2004.